Proceedings of the 2004 Winter Simulation Conference
R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, eds.

A METHODOLOGICAL FRAMEWORK FOR BUSINESS-ORIENTED MODELING OF IT INFRASTRUCTURE

Ariel Landau
Segev Wasserkrug
Dagan Gilat
Natalia Razinkov
Aviad Sela
Sarel Aiber

IBM Haifa Research Lab
University Campus
Carmel Mountains
Haifa, 31905, ISRAEL

ABSTRACT

The creation of IT simulation models for uses such as capacity planning and optimization is becoming more and more widespread. Traditionally, the creation of such models required deep modeling and/or programming expertise, thus severely limiting their extensive use. Moreover, many modern intelligent tools now require simulation models in order to carry out their function. For these tools to be widely deployable, the derivation of simulation models must be made possible without requiring excessive technical knowledge. Hence we introduce a general methodology that enables an almost automatic deployment of IT simulation models, based on three fundamental principles: Modeling only at the required level of detail; modeling standard components using pre-prepared models; and automatically deriving the application-specific model details. The technical details underlying this approach are presented. In addition, a case study, showing the application of this methodology to an eCommerce site, demonstrates the applicability of this approach.

1 INTRODUCTION

The requirement for simulation models for tasks like capacity planning or optimization, and in particular on-demand self-optimization, is becoming more and more widespread. The commercial market nowadays offers a wide range of software tools for the creation of performance simulation models (CSIM 19, Arena, AnyLogic, Simulink, etc.). However, a would-be-creator of these models is faced with the fact that the creation of (non-trivial) simulation models on top of these tools often requires quite a deep knowledge in both performance modeling and computer programming. In addition, even when the expertise required for the creation of a simulation model is available, there is quite often still a need to provide the model with extensive data, specific to the environment being modeled. Experience shows that such data is in most cases extremely difficult to retrieve from actual environments.

The above two shortcomings in the creation of simulation models severely limit their widespread applicability. In addition, there are currently several intelligent tools (e.g., Aiber et al. 2004, Levy et al. 2003) that rely on performance simulation models to perform their function.

For example, Aiber et al. (2004) introduces a tool whose purpose is to enable the configuration of an IT site so as to optimize business objectives. This task is carried out by implementing the architecture depicted in Figure 1 below.

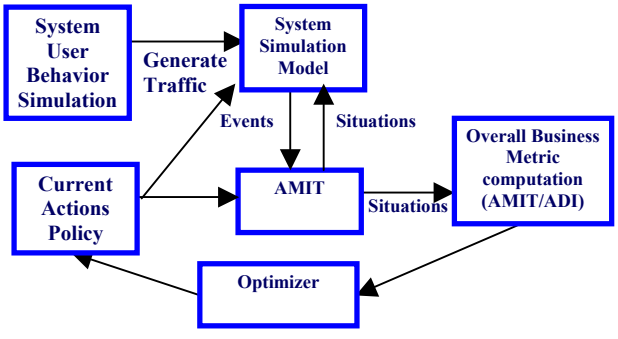

Figure 1: The Business-Objective Driven Optimization Architecture

The above architecture defines the following process: The IT system is simulated, taking into account both the behavior of system resources (i.e., applications, middleware, hardware), the incoming traffic (i.e., client requests),

and a set of configuration parameters that are subject to optimization. This simulation is then coupled with the AMIT/ADI rule engine (Adi and Etzion 2002, Adi et al. 2003), for the calculation of the business results (for each given configuration), and an optimizer, that searches for the best possible configuration. For more details see Aiber et al. (2004).

Tools such as the one outlined above, for which performance simulation models are mandatory, will not be widely deployable unless a way is found to produce the required performance simulation models without requiring too much modeling or programming knowledge on the side of the deployer.

In this paper, a general methodology is introduced, aimed at enabling a simple, non-programmatic, and almost automatic generation and deployment of IT simulation models, with particular emphasis on eCommerce environments. In addition, a case study showing the application of this methodology to an eCommerce site, demonstrates the value and applicability of this approach.

The remainder of this paper is organized as follows: Section 2 briefly reviews some related work. In Section 3, an outline of the proposed methodology is given, followed by a detailed technical discussion of the model creation process. Section 4 presents a modeling and validation case study, where the concepts discussed in the paper are illustrated. Finally, the article concludes with a summary in section 5.

2 RELATED WORK

As mentioned in the introduction, there are several commercially available products for the construction of general purpose performance simulation models (CSIM 19, Arena, AnyLogic, Simulink, etc.). The development of IT simulation models using these products usually requires extensive modeling and programming knowledge. In addition, issues related to the derivation of application-specific information are beyond their scope. Some modeling tools, such as HyPerformix IPS Optimizer, include pre-prepared models, thus reducing the amount of modeling expertise required.

As for algorithms for automatic derivation of application-specific models, we can mention the *Customer Behavior Model Graph,* or CBMG, method (Menascé and Almeida 2000), which defines a process for the automatic derivation of website client behavior models out of website access logs.

3 THE MODELING METHODOLOGY

The present section introduces a modeling methodology that enables a non-programmatic and almost automatic creation and deployment of IT simulation models. The methodology is based on the following three principles:

- **Modeling only at the required level of detail**: Depending on the actual use, models containing different levels of detail may suffice. It would be highly desirable that generated models include all the necessary details, but not more than the required detail. For example, one of the chief applications for the models created following our methodology is the assessment of the impact of decisions at the IT level on certain pre-defined objectives, and in particular, business objectives, There may be cases in which a detailed model of a specific middleware or hardware, though present in the modeled environment may not be required. Therefore, pre-prepared model components, or *building blocks* (to be discussed in greater detail in the next paragraph), may be implemented at several levels of detail, ranging from implementations containing very detailed queuing and resource-usage models, to "black box", or functional, models.
- **Creating pre-prepared models of standard components**: To enable the creation of models of IT topologies in a simple, non-programmatic manner, given an IT site, standard components of the IT (e.g., middleware, operating systems and hardware) are modeled using a set of pre-prepared building blocks. Following this approach, for each type of IT component, a generic model is created in advance, which can be reused across different sites and infrastructures. It's worth noting that since we attempt to create models containing only the required level of detail, a single building block may happen to contain several tiers of a system, and in some cases, contain all the standard components of the modeled system.
- **Automatically deriving application-specific models**: Models of IT topologies, created by assembling the building blocks described in the previous paragraph, model only hardware resources and standard middleware, and do not contain any information about particular applications running on top of this middleware. In addition, unlike a description of the IT topology, which in many cases can be obtained from human sources, this performance-related application-dependent information is usually not explicitly known. For this reason, the methodology defines a set of high-level application-specific model specifications. Models satisfying these specifications are generated automatically, using machine learning algorithms and statistical techniques.

The modeling methodology also defines a two-stage process for the generation of IT simulation models: A first stage, where a **topological IT model** is built by assembling building blocks (e.g., using a drag-and-drop GUI), and a second stage, consisting of the deployment of application-specific automatic model-learning algorithms. The models

required for both stages, namely pre-prepared building blocks and application-specific models, will be next discussed in detail.

3.1 Pre-Prepared Building Blocks

Pre-prepared building blocks are re-usable models of hardware resources and standard middleware that are coded in advance, and are used in the assembly of topological IT models.

From a conceptual point of view, we have divided pre-prepared building blocks into several categories. This is mainly motivated by the intention to ease the process of model creation and management. We discriminate between blocks that model aspects of the actual system, and blocks that, while not representing any objects in the system, facilitate the connection of the model to the external world. Blocks of the first kind are further classified into blocks that model *physical* behavior, and those that model *logical* (or algorithmic) behavior. They match the "**System Simulation Model**" and "**System User Behavior Simulation"** components of the architecture depicted in Figure 1 above. Other blocks include those that collect and process information generated by a simulation, thereby matching the "**AMIT**" and "**Overall Business Metric Computation**" components in Figure 1, and those that act on the model, e.g., update the IT topology, configure policy parameters, etc., which correspond to the "**Current Action Policy"** and "**Optimizer**" components in Figure 1.

Having this in mind, we have defined the following building-block categories:

- **Core infrastructure**: These blocks model the cost (in terms of time spent in the system) of performing operations in the modeled environment, e.g., performing a database request, serving an HTTP at a J2EE application server, etc. Examples of core-infrastructure building blocks include hardware resource models, such as CPUs and disks, and middleware models, e.g., models of web, application, or database servers.
- **Environmental modifiers**: These blocks intend to model logical features of an actual environment, such as routing, load-balancing or dynamic bandwidth-allocation policies, so that their impact on the system's overall performance can be ultimately assessed. They are also used to model objects that are essential to the environment, but are not part of the IT environment, e.g., an input model, or a client-traffic generator. Unlike core-infrastructure blocks, environmental modifiers intend to capture the functional behavior of the modeled objects, disregarding the performance cost of going through any system resources.
- **Monitors**: These blocks interact with other model objects through an instrumentation interface. They receive primitive modeled events, such as the start or the completion of a client transaction, perhaps together with additional information about the transaction, such as the transaction's amount where applicable, or the identity of the client performing the transaction. Monitors act as translators between these primitive events, and business objectives such as the operating profit generated by an eCommerce website. To this end, a monitor will typically implement, or be a wrapper of, a situation manager, or a complex-event-processing or business-rule engine.
- **Actors**: These blocks modify configurable parameters of model objects, such as those parameters on which an optimization is based, or the whole model, e.g., through the creation or removal of model components, connections, etc.

3.2 Automatically-Created Application-Specific Models

The automatically-created application-specific models provide site-specific information required by the IT topological models, defined above.

We have defined the following application-specific models: the user-behavior model, the user-attribute model, the tier-level message breakdown, and tier-specific resource-requirement models. They were created in the context of models for an eCommerce website. However, they can be easily adapted to other domains (e.g., messaging applications). The models, as well as the automatic algorithms used to derive them, will be now discussed in detail.

3.2.1 User Behavior Model

The user-behavior model is a model of traffic patterns of client requests submitted to an eCommerce website. These patterns are modeled using a set of Markovian state-transition graphs, augmented with a set of client clusters.

The derivation of the Markovian transition graphs is based on the *Customer Behavior Model Graph* (CBMG) method (Menascé and Almeida 2000). This method assumes the existence of a finite number of known client-request types, and that actual client requests can be naturally grouped into sessions. Sessions are then interpreted as traversals of a Markovian state-transition graph whose nodes are client-request types. The definition of a *distance* between sessions permits the application of standard clustering algorithms, such as the *k*-means clustering algorithm (Everitt 1980), yielding a small number of state-transition graphs. These graphs are seen as representing different *client-session types.* Figure 2 below shows an example of such a transition graph.

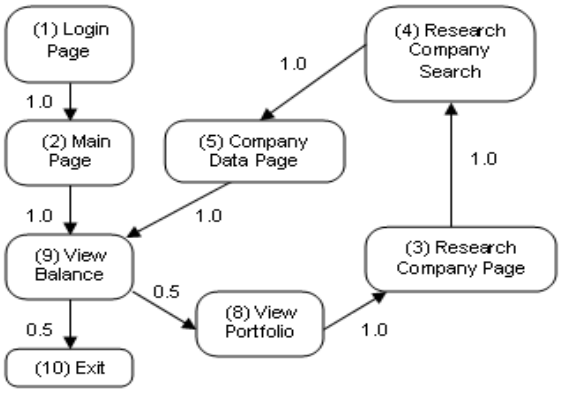

Figure 2: Example of a CBMG, for a Given Session Type

The user-behavior model then groups simulated clients into clusters, according to the frequency with which each of them initiates sessions belonging to the different client-session types. This clustering process yields a set of probability distributions (having the set of client-session types as their sample space), and a function that assigns one of these distributions to each simulated client.

The user-behavior model is automatically derived by applying the CBMG derivation algorithm (Menascé and Almeida 2000) and *k*-means clustering algorithm (Everitt 1980) to web logs of the modeled environment.

3.2.2 User Attribute Model

The user-attribute model captures data that accompanies each incoming client request, which is relevant to the computation of business objectives. An example of this is the actual amount of a purchase transaction. Unlike the user-behavior model, which models the arrival process of client requests, the user-attribute model deals with parameters of these client requests on which the rules for computing the business objectives are based.

The model is composed of a set of client clusters, and, for each cluster, a set of attribute-generating probability distributions. Clients are clustered according to similarity of attributes, where similarity means closeness of probability distributions. Standard statistical tests such as chi-squared can thus be applied for measuring this similarity. Table 1 below shows an example of a user-attribute model, based on three abstract attributes (A, B, and C), and composed of three clusters (C1, C2, and C3).

Model derivation is based on standard automatic distribution fitting and clustering algorithms (Brownlee 1965, Everitt 1980). First, for each simulated client, a probability distribution for each relevant attribute is derived from actual measured data. Simulated clients are then clustered according to the distance between these probability distributions, as measured by some standard statistical similarity test such as chi-squared. The centroid of each cluster is the set of attribute-generating probability distributions that is attached to it.

Table 1: Example of a User Attribute Model

C1	A=20 to 30 Unif. distr.	B= Man/Woman (50%/50%)	C='gold'
C2	A=Normal μ=50 σ=10	B= Man/Woman (40%/60%)	C='platinum'
C3	A=20 to 40 Unif. distr.	B= Man/Woman (65%/35%)	C='gold' /'regular' (50%/50%)

Relevant user attributes usually appear as part of HTTP client requests. The user-attribute model is automatically generated from logs and database values on the customer's site, following the process just described.

3.2.3 Tier-Level Message Breakdown Model

The tier-level message breakdown model breaks down incoming client requests (as modeled by the user-behavior model), into invocations of methods and services at the different components of a multi-tier web application, e.g., in the context of a J2EE web application, Servlets, JSPs, EJBs and database requests.

The tier-level message breakdown of each client-request type is modeled by a probabilistic graph, whose nodes are services and methods that are invoked. Figure 3 below shows an example of such a breakdown graph.

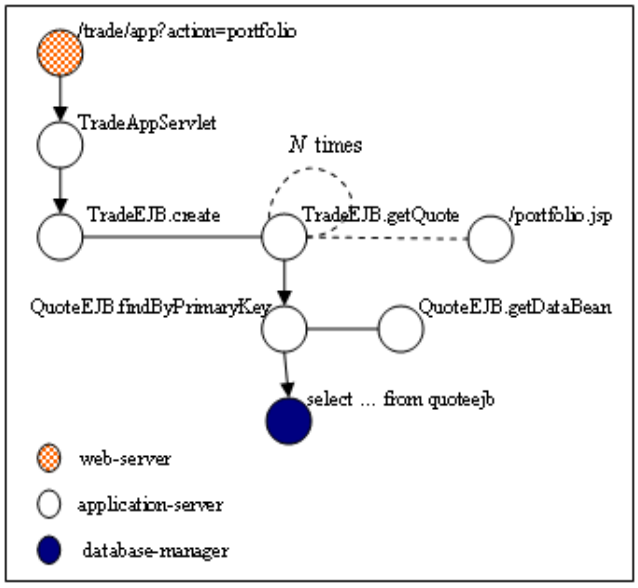

Figure 3: Example of a Tier-Level Message Breakdown Model Graph

In this graph, each node represents a method call; different colors and shades are used to represent nodes of different system tiers. Vertical edges represent *caller/callee flows*, i.e., an invocation of a method within another method, while horizontal edges represent *consecutive flows*, i.e., a sequence of method invocations that take place in a specified order.

Solid edges in the tier-level message breakdown graph represent deterministic edges, while dotted lines represent probabilistic edges. In the particular example of Figure 3, the dotted circular line, which is a recursive sequential call, is supplied with the random variable *N*, that establishes the number of loop iterations, i.e., how many times this edge is to be taken. The other dotted line is the "default edge", namely, the edge that is taken if no other edge (exiting from the same node) is taken.

Breakdowns of individual client request are collected by tracing and monitoring tools (Intel's VTune Enterprise Analyzer, IBM's Tivoli Monitoring for Transaction Performance). Tier-level message breakdown graphs are then built by applying business-process analysis (Golani and Pinter 2003) and distribution fitting (Brownlee 1965) algorithms to a log of individual broken-down client requests.

3.2.4 Tier-Specific Resource Requirement Model

The tier-specific resource-requirement models are models of the resource level (e.g. CPU, disk) service times (i.e. the net usage time of each resource) of each tier-level request.

For tiers where a detailed queuing-network building block has been implemented, service times required at individual resources, for the completion of each tier-level service-request type, are first measured at the website in isolation, using monitoring tools such as Intel's VTune Enterprise Analyzer or IBM's Tivoli Monitoring for Transaction Performance. Distribution functions for these service times are then automatically generated by applying statistical fitting methods (Brownlee 1965). The resource-requirement model so obtained is then tested and validated against real website loads. If necessary, load-dependent corrections, or *burstiness factors* (Menascé and Almeida 2002), are introduced.

For tiers where a black-box building block is considered sufficient, Bayesian-network learning algorithms (Ghahramani 1998) are applied, to derive a response time function for each request type, depending on the state of the modeled component. Our Bayesian-network learning algorithm assumes that the state of a modeled component changes only when a request is received, or when a request has been serviced and is returned. The component's state is therefore uniquely defined by the set of pending (i.e., received and not yet returned) requests. The algorithm receives as input an entry/exit log of requests to the modeled tier, and produces two functions:

- A response-time function $f(R, \underline{\theta})$, which depends on the state $\underline{\theta}$ of the modeled component, and for each pending request R, returns the time that remains till the request is to be returned.
- A state-transition function $g(R, \underline{\theta})$, that computes the new machine-state $\underline{\theta}'$ of the modeled component as a function of the previous machine-state $\underline{\theta}$, and the request R, that either has just arrived, or has been just returned.

4 A CASE STUDY – MODELING AND VALIDATING A SPECIFIC ENVIRONMENT

This section details the application of our methodology to the modeling and validation of a specific eCommerce site. Our example is based on a three-tier deployment of the FMStocks trading application.

As the final goal of our model is the optimization of the IT environment according to business objectives, it is our aim to validate the model against business objectives rather than against IT metrics. Since business objectives are, however, based on IT metrics, the validation of the model against IT metrics does provide valuable information. On the other hand, whenever certain IT metrics prove to have a negligible impact on business objectives, their validation becomes irrelevant.

The exposition starts with a description of the actual IT environment, the input model used, and the business rules applied. Then, the two-stage model-generation process is described: (1) the assembling of an IT model out of building blocks, including a detailed description of the building blocks used, and (2) the derivation of application-specific models. The section ends with a validation of the simulation model, and a set of conclusions.

4.1 The Actual Environment

The actual environment of our case study is a website running the FMStocks 2000 application (available online at <http://www.fmstocks.com>). It consists of three interconnected computers running the IIS web-server, the COM+ application server, and the SQLServer database server, see Figure 4 below. Client traffic was emulated, according to the input model described in 4.2

Intel's VTune Enterprise Analyzer was used to measure roundtrip response times at the web, application and database tiers.

4.2 Input Model and Business Metrics

The input model consists of a fixed number of simulated clients; each of them initiates, on average, two sessions on every 8-hour working day. Thus, for each 1,000 simulated clients, there's a workload of 250 sessions per hour. The session-arrival process is a Poisson process, with the appropriate intensity.

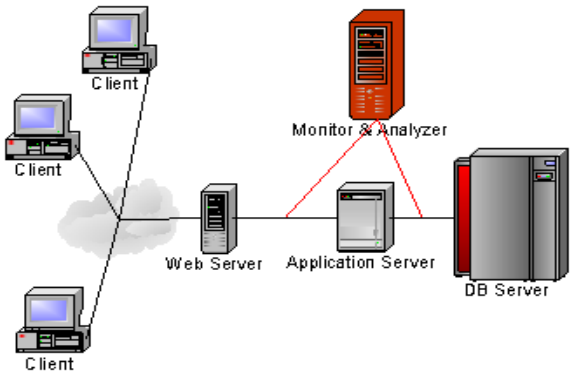

Figure 4: The Actual Environment

Four client-session types were defined, and a probability assigned to each of them: *Browse/Buy* (33%), *Browse* (28%), *Check Portfolio* (22%), and *Buy* (17%). We have defined each session type as a Markovian transition graph. As an illustration, Figure 5 below shows the graph corresponding to a *Browse/Buy* session (this is an expanded version of the example shown above in Figure2).

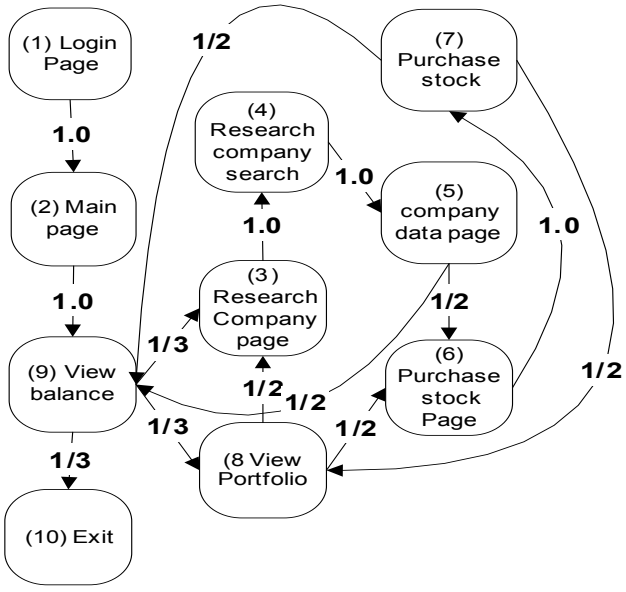

Figure 5: The CMBG Corresponding to the *Browse/Buy* Session Type

User attributes were defined as follows:

- Service Level Agreement (SLA): Platinum (10% of the clients), gold (40%), and regular (50%).
- Spending Habits: High spenders (monthly spending amount uniformly distributed between $100,000 and $125,000 per month, 3% of the clients), medium spenders ($10,000 to $12,500, 30%), and low spenders ($1,000 to $1,250, 67%).

The *client think-time*, that is, the time elapsed between the reply from a request and the submission of a new request by the same client (within a single session), is uniformly distributed between 0.5 and 8 seconds. Finally, if a request is not answered within a 10-second timeout, the client session is aborted.

The computation of the business metric is governed by the following rules:

- For each buy or sell transaction, a commission of 4% of the transaction's amount, or $25 is collected, whichever is greater.
- Platinum and gold customers pay a daily flat fee of $50 and $20, respectively.
- Platinum and gold customers are promised an average response time of one second, and two seconds, respectively. For each 1% deviation from this contract, the site pays them a penalty of $5.5 and $3.5 respectively.

4.3 The IT Simulation Model

Following our two-stage methodology, the generation of an IT simulation model for the actual environment consisted of (1) assembling building blocks into a model of the IT topology, followed by (2) a stage where the model is complemented with application-specific information. These two stages will be discussed below.

4.3.1 Building Blocks and IT Topology

The composition of the simulation IT topology model was based on the following building blocks:

- A CPU building block, implementing a round-robin queue.
- A disk building block, implementing a first-come-first-served queue.
- A generic middleware building block that translates messages it receives from its front end into messages for a CPU, a disk and/or a lower tier (the translation is to be specified by application-specific models, such as the tier-level message breakdown model, and the tier-specific resource requirement model).
- A traffic generator building block that simulates the input model described in the previous subsection.

In addition, response times were monitored at each of the middleware building blocks, and business metrics computed following the rules specified above.

The IT simulation model was created by assembling instances of these building blocks, using XJTek's AnyLogic 4.5 (<http://www.xjtek.com/anylogic>) simulation product. The model consisted of one traffic

generator block, and three sets of middleware/CPU/disk blocks, representing the web, application and database servers respectively. The simulation model is shown in Figure 6. The upper diagram shows a model of the whole system, while the lower diagram shows the internal structure of generic middleware building block.

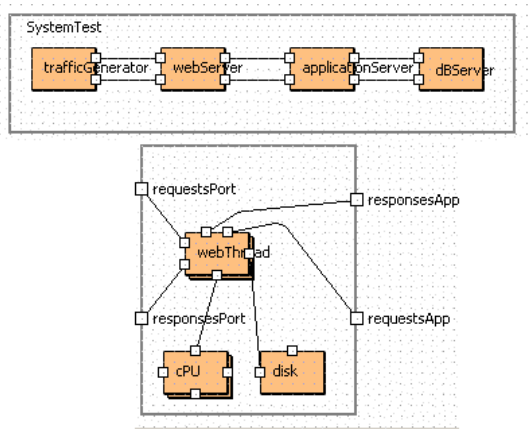

Figure 6: The IT Simulation Model

4.3.2 The Application-Specific Models

A tier-level message breakdown model and a tier-specific resource requirement model were derived from the actual environment. Since the actual environment was not a real business environment, the user-behavior model, and the user-attribute model could not be derived; these models were in fact provided as input to the environment.

Both application-specific models were derived by running isolated client sessions, consisting of a fixed sequence of client requests, and using Intel's VTune Enterprise Analyzer to monitor the system's activity. As sessions were run in isolation, response times were interpreted, as a first approximation, as actual resource requirements. A correction procedure aimed at dealing with contentions not accounted for in our IT topology model was then applied (more details regarding this technique will be given in 3.5 below).

Table 2 shows some results of these model derivations. The left side of the table shows average values of some tier-specific resource-requirement probability distributions. Resource requirements for the web server and application server tiers are CPU average service times, in milliseconds (no significant disk activity was observed for these tiers). For the database server tier, there are two columns, showing CPU and disk average service times respectively.

The right side of the table shows part of the tier-level message breakdown: Each of the five rightmost columns represents a task at the web tier. Lower-tier tasks invoked as a result of the processing of a web-tier task are marked with an 'X' inside the appropriate cell.

Table 2: Average Resource Requirements, and Tier-Level Message Breakdowns

Web Tier	**Time**	LP	L	VB	BS	VP
Login Page	9	X				
Login	114		X			
View Balance	15			X		
Buy Stock	22				X	
View Portfolio	20					X

App Tier	**Time (ms)**	L	VB	BS	VP
Acct.Summary	35		X		
Acct.ListPostns	27				X
Acct.VfyLogin	32	X			
Broker.BuyStock	110			X	
Ticker.VfySymb	17			X	

DB Tier	**Time (ms)**		L	VB	BS	VP
Acct Summary	0.93	0.01		X		
Acct VfyLogin	0.00	0.00	X			
Broker Buy	0.94	0.09			X	
Position List	3.31	0.97				X
Ticker Price	0.31	0.02			X	
Ticker VfySymb	0.35	0.02			X	
Xactn Order	0.35	0.02			X	
Xactn Type	0.00	0.00			X	

4.4 Validation and Tuning Against IT Metrics

Response-time results from the simulation model and the actual environment were compared, for the input model defined above, at various workload intensities ranging from 250 to 2000 sessions per hour (a throughput of 2500 sessions per hour was found to be beyond the system's capacity).

Response times for the database tier turned out to be comparatively very small, erratic, and highly volatile, which led us to the (expected) conclusion that, for our particular environment, we could have ignored the effect of the database tier at all in our simulation model.

For the web server and the application server, it was found that the model overestimates response times for relatively low intensities, and underestimates response times for high intensities. In addition, this gap was found to increase together with the intensity. These were attributed, on one hand, to the synergetic effects of low, though steady, input on factors such as caching effects, and on the other, to burstiness and contention effects not explicitly accounted for in our IT topology model. Following the methodology developed in (Menascé and Almeida 2002), we introduced two correction factors aimed at closing these gaps: a negative (i.e., smaller than one) constant factor for the synergy resulting from having a steady load, and a positive (i.e., larger than one) burstiness factor, proportional to the workload intensity.

Table 3 below, shows a partial list of corrective factors used. The "synergy" factor is constant (i.e., it doesn't depend on workload intensity). The tabulated "burstiness" factor is the correction applied to a workload intensity of 250 sessions per hour.

Table 3: Corrective Factors

Tier – Task	Synergy	Burstiness
WEB – Login Page	-20%	+2%
WEB – Login	-	+5%
WEB – View Balance	-	+5%
WEB – Buy Stock	-	+10%
WEB – View Portfolio	-	+10%
APP – Acct.Summary	-	+4%
APP – Acct.ListPostns	-15%	+5%
APP – Acct.VfyLogin	-	+4%
APP – Broker.BuyStock	-	+4%
APP – Ticker.VfySymb	-	+4%

Figures 7 and 8 show some response time gaps between the model and the actual environment, before and after the burstiness correction, for a subset of the tasks listed in Table 3. A positive (negative) gap indicates that model response times overestimate (underestimate) those of the real environment.

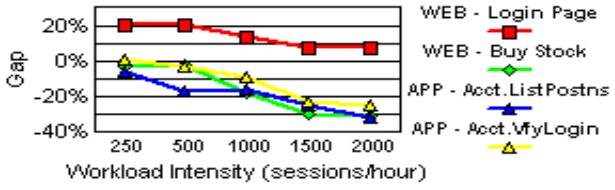

Figure 7: Gaps Between Model and Real Environment Response Times Before Correction

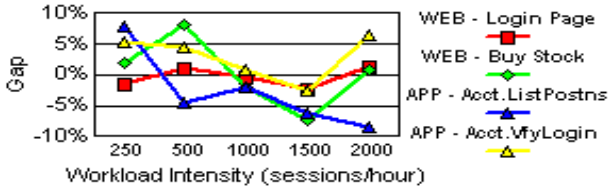

Figure 8: Gaps Between Model and Real Environment Response Times After Correction

The final validation of the simulation model was carried out by comparing the overall business metric, as obtained both from the actual environment and from the simulation model. Figure 9 below shows collected-commission gaps between the actual environment and the corrected simulation model. The result shown in each case is the largest gap observed in a set of three independent measurements (again, positive gaps indicate that the simulation model overestimates actual results).

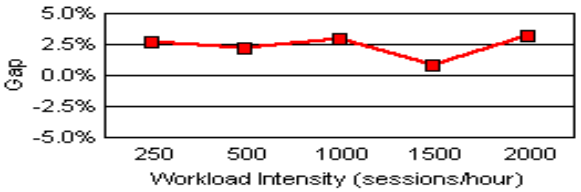

Figure 9: Gaps Between Model and Real Environment Collected Commission (After Correction)

4.5 Conclusions

The validation process has shown that a gap of no more than 10% exists between our simulation model and our actual environment, for response times at the web-server and application-server tier, and for business metrics. No validation was achieved for response times at the database-server tier. Nevertheless, it was observed that the database-server has negligible impact on business metrics, and therefore, for the defined business objectives, the database tier might have been ignored.

5 SUMMARY

This paper presented a methodology and a process for business-oriented performance modeling of IT infrastructures. The methodology has many potential applications, ranging from traditional capacity planning to business-objective driven optimization that focus on high-level business objectives, rather than more traditional IT metrics.

Our methodology is quite general, and may be applied to various IT scenarios, including eCommerce sites and messaging infrastructures. In addition, business objectives may be defined in an arbitrarily complex manner, which in turn helps maintaining a clear connection between IT policy decisions and business-level metrics, such as operating profit or return on investment. A two-stage process enables an almost automatic deployment of the modeling solution, given an IT site. The knowledge on the IT site required for an actual deployment is minimal.

While the first stage of the process, dealing with composition of topological models out of pre-prepared building blocks, is domain-agnostic, deciding what sets of building blocks are suitable for pre-preparation in different application domains, as well as deciding how much detailed should be a given building block, are issues that warrant further research. In contrast, the second stage of the process, which deals with the extraction of application-specific information, is more domain-specific. The paper presented our choice of models in the context of eCommerce sites. These models can be easily adapted to related domains, such as messaging applications. Processes for defining sets of application-specific models in more dissimilar domains are worth of being explored.

ACKNOWLEDGMENTS

The authors wish to acknowledge the contribution of Nathalie Sznajder to the development of the user-input models for the case study presented in this paper.

REFERENCES

Adi, A., and O. Etzion. 2002. The Situation Manager Rule Language. In *Proceedings of the International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, ed. M. Schroeder, and G. Wagner. *CEUR Workshop Proceedings* 60: 36-57.

Adi A., O. Etzion, D. Gilat, G. Sharon. 2003. Inference of Reactive Rules from Dependency Models. In *Rules and Rule Markup Languages for the Semantic Web*, ed. M. Schroeder, and G. Wagner, 49-64. Lecture Notes in Computer Science 2876, Springer.

Aiber, S., D. Gilat, A. Landau, N. Razinkov, A. Sela, and S. Wasserkrug. 2004. Autonomic Self-Optimization According to Business Objectives. In *First International Conference on Autonomic Computing,* 206-213. IEEE Computer Society.

Brownlee, K. A. 1965. *Statistical Theory and Methodology in Science and Engineering.* 2nd ed. New York: John Wiley & Sons.

Everitt, B. S. 1980. *Cluster Analysis*. 2nd ed. New York: Halsted Press.

Ghahramani, Z. 1998. Learning Dynamic Bayesian Networks. In *Adaptive Processing of Sequences and Data Structures,* ed. C. L. Giles, and M. Gori, 168-197. Lecture Notes in Computer Science 1387, Springer.

Golani, M., and S. S. Pinter. 2003. Generating a Process Model from an Audit Log, *Business Process Management* 2003: 136-151.

Levy, R. M., J. Nagarajarao, G. Pacifici, M. Spreitzer, A. N. Tantawi, A. Youssef. 2003. Performance Management for Cluster Based Web Services. In *Integrated Network Management VIII*, ed. G. S. Goldszmidt, and J. Schönwälder, 247-261. Kluwer Academic Publishers.

Menascé, D. A., and V. A. Almeida. 2000. *Scaling for E-Business*. Upper Saddle River, New Jersey: Prentice Hall PTR.

Menascé, D. A., and V. A. Almeida. 2002. *Capacity Planning for Web Services*. Upper Saddle River, New Jersey: Prentice Hall PTR.

AUTHOR BIOGRAPHIES

ARIEL LANDAU is a research staff member in the IBM Haifa Research Lab, Israel. Since joining IBM in 1997, he has been working on performance modeling, performance monitoring, and dynamic instrumentation projects. He has an MSc. in Mathematics from the Technion, Israel Institute of Technology (1997). His e-mail address is <ariel@il.ibm.com>.

SEGEV WASSERKRUG is a research staff member in the Active Solutions group of the IBM Haifa Research Lab, Israel. He leads the development of technologies for optimizing IT configurations according to business objectives, using hybrid simulation models, advanced optimization techniques, and machine learning algorithms. He has an MSc. in Computer Science, and is studying towards his Ph.D. in Information Systems Engineering at the Technion, Israel Institute of Technology. His e-mail address is <segevw@il.ibm.com>.

DAGAN GILAT, Ph. D., is the Manager of the Active Solutions group in the IBM Haifa Research Lab, Israel. He has over 15 years experience in technology research and development. Previous to IBM, he worked as a consultant for private and public Israeli companies. His main interests include simulation, modeling, and advanced internet technologies. Dr. Gilat studied at the Technion, Israel Institute of Technology, where he received his Ph.D. in Information Systems, MSc. in Operations Research, and BA in Computer Science. His e-mail address is <dagang@il.ibm.com>.

NATALIA RAZINKOV is a research staff member in the IBM Haifa Research Lab, Israel. Since joining IBM in 2000, she has been working on active decision making projects, and performance modeling and monitoring. She has an MSc. in Computer Science (1995) and a Ph.D. in Automated Control Systems and Advanced Information Technologies (1999), both from the National Aerospace University "Kharkov Aviation Institute", Ukraine. Her e-mail address is <natali@il.ibm.com>.

AVIAD SELA is a research staff member in the IBM Haifa Research Lab, Israel. Since joining IBM in 1997, he has been working on speech recognition, signal processing, and performance modeling projects. He has a BSc. in Mechanical Engineering and a BSc. in Computer Science, both from the Technion, Israel Institute of Technology. His e-mail address is <sela@il.ibm.com>.

SAREL AIBER is a research staff member in the IBM Haifa Research Lab, Israel. Since joining IBM in 2002, he has been working on performance modeling, change detection and adaptation in a business environment, and business rule optimization. He has a BSc. with honors in Computer Science from the University of New South Wales. His honors thesis is in the area of machine learning. His e-mail address is <aiber@il.ibm.com>.